Package ‘Brundle’

September 18, 2017

Type Package

Title Brundle: Tools for the normalisation of ChIP-Seq data

Version 1.0.0

Author Andrew Holding

Maintainer Andrew Holding <andrew.holding@cruk.cam.ac.uk>

Description Brundle provides a series of convienicе functions for normalising ChIP-seq data. This input can either be from DiffBind or a matrix formatted for DESeq2. The output is either a Diffbind object or the defualt DESeq2 output. Either can then be processed as normal.

License CC BY 4.0

Encoding UTF-8

LazyData true

Depends DiffBind, Rsamtools, DESeq2, lattice

RoxygenNote 6.0.1.9000

biocViews Software, Technology, Sequencing, ChIPSeq

R topics documented:

`jg.applyNormalisation` *jg.applyNormalisation*

Description

Takes the experimental peakset and applies the calculated coefficient and correction factor.

Usage

```
jg.applyNormalisation(jg.experimentPeakset, jg.coefficient, jg.correctionFactor,
  jg.treatedNames)
```

Arguments

`jg.experimentPeakset`	is the peakset extracted from the Diffbind object
`jg.coefficient`	is the coefficient calculated by jg.getNormalizationCoefficient
`jg.correctionFactor`	is the correction factor calculated by jg.getCorrectionFactor
`jg.treatedNames`	is the names of the treated samples

Examples

```
data(jg.experimentPeakset, package="Brundle")
jg.experimentPeaksetNormalised<-jg.applyNormalisation(jg.experimentPeakset, 1.267618, 0.6616886, c("1b", "2
```

`jg.convertPeakset` *jg.convertPeakset*

Description

Converts a DiffBind object into a DESeq2 compatible form for the workflow.

Usage

```
jg.convertPeakset(jg.controlPeakset)
```

Arguments

`jg.controlPeakset`	is the name of the DiffBind object to convert

Examples

```
jg.convertPeakset(jg.controlPeakset)
```

`jg.correctDBASizeFactors` *jg.correctDBASizeFactors*

Description

Correct the size factors in a DiffBind object using our DESeq2 pipeline for normalisation.

Usage

```
jg.correctDBASizeFactors(dba, jg.controlSizeFactors)
```

Examples

```
jg.correctDBASizeFactors(dba,jg.controlSizeFactors)
```

`jg.countAlignedMReads` *jg.countAlignedMReads*

Description

This function counts the number of aligned reads in millions from a list of bam files. It returns these in as a list of numbers in the same order.

Usage

```
jg.countAlignedMReads(jg.bamFiles)
```

Arguments

`jg.bamFiles`	is a list of bam files to count.

`jg.dbaGetPeakset` *dbaGetPeakset*

Description

Extracts a peakset from a dba object.

Usage

```
jg.dbaGetPeakset(dba)
```

Arguments

`dba`	is the name of the DiffBind object

Examples

```
data(dbaExperiment, package="Brundle")
jg.experimentPeakset <- jg.dbaGetPeakset(dbaExperiment)
```

`jg.getControlCounts` *jg.getControlCounts*

Description

This function counts the number of aligned reads in millions from a list of bam files. It returns these in as a list of numbers in the same order.

Usage

```
jg.getControlCounts(jg.control, jg.controlSampleSheet, jg.Condition)
```

Arguments

`jg.control`	is a peakset extracted by jg.dbaGetPeakset.
`jg.controlSampleSheet`	is the samplesheet supplied to DiffBind to generate jg.control.
`jg.Condition`	is the condition we the counts for as specficied in the samplesheet.

Examples

```
data(jg.controlPeakset, package="Brundle")
fpath <- system.file("extdata", "samplesheet_SLX14438_hs_CTCF_DBA.csv",package="Brundle")
jg.controlSampleSheet<-fpath
jg.controlCountsTreated<-jg.getControlCounts(jg.controlPeakset, jg.controlSampleSheet,"Fulvestrant")
```

`jg.getCorrectionFactor` *jg.getCorrectionFactor*

Description

Generates a correction factor that is applied before reinserting the data into the DiffBind object for analysis.

Usage

```
jg.getCorrectionFactor(jg.experimentSampleSheet, jg.treatedNames,
  jg.untreatedNames)
```

Arguments

`jg.experimentSampleSheet`	is the csv samplesheet used to load the data into DiffBind
`jg.treatedNames`	is a list of the names of samples that are treated
`jg.untreatedNames`	is a list of the names of samples that are untreated

Examples

```
data(jg.controlCountsTreated, package="Brundle")
data(jg.controlCountsUntreated, package="Brundle")
jg.coefficient<-jg.getNormalizationCoefficient(jg.controlCountsTreated, jg.controlCountsUntreated)
```

`jg.getDba` *jg.getDba*

Description

Generates a DiffBind object from a valid SampleSheet with the required data for normalisation. No examples are provided as BAM files are not included in this package.

Usage

```
jg.getDba(jg.experimentSampleSheet, dbaSummits, ...)
```

Arguments

`jg.experimentSampleSheet`	is the filename of samplesheet to be loaded
`dbaSummits`	is the peak width in bp from summits (optional)

`jg.getNormalizationCoefficient` *jg.getNormalizationCoefficient*

Description

This function allows the user to caryy out the normalisation and returns a coefficient by using a linear fit to the control data.

Usage

```
jg.getNormalizationCoefficient(jg.controlCountsTreated,
  jg.controlCountsUntreated)
```

Arguments

`jg.controlCountsTreated`	Control counts extracted from the Diffbind object for the treated condition using jg.getControlCounts
`jg.controlCountsUntreated`	Control ounts extracted from the Diffbind object for the untreated condition using jg.getControlCounts

Examples

```
data(jg.controlCountsTreated, package="Brundle")
data(jg.controlCountsUntreated, package="Brundle")
jg.coefficient<-jg.getNormalizationCoefficient(jg.controlCountsTreated,
                                              jg.controlCountsUntreated)
```

`jg.getSampleIds` *jg.getSampleIds*

Description

Extracts the sample Id from DiffBind formatted SampleSheet in csv format.

Usage

```
jg.getSampleIds(jg.controlSampleSheet)
```

Arguments

`jg.controlSampleSheet`	is the filename of the samplesheet

`jg.plotDeSeq` *jg.plotDeSeq*

Description

Plots the output from DESeq2 for the Brundle pipeline

Usage

```
jg.plotDeSeq(ma.df, p = 0.01, title.main = "Differential ChIP",
  log2fold = 0.5, flip = FALSE)
```

Arguments

`ma.df`	is the result Dataframe from DESeq2
`p`	is the minium FDR to highlight as significant
`log2fold`	is the minimum log2 fold change for highlighted points
`flip`	when set to TRUE flips the data
`title`	is the plot title

Examples

```
data(jg.experimentResultsDeseq,package="Brundle")
jg.plotDeSeq(jg.experimentResultsDeseq,
 p=0.01,
 title.main="Fold-change in ER binding",
 flip=TRUE
)
```

`jg.plotDeSeqCombined` *jg.plotDeSeqCombined*

Description

Overlays the plots from the output from DESeq2 for the Brundle pipeline

Usage

```
jg.plotDeSeqCombined(jg.controlResultsDeseq, jg.experimentResultsDeseq,
  title.main, padjX, flip = FALSE)
```

Arguments

`jg.controlResultsDeseq`	is the result Dataframe from DESeq2 for the control conditions
`title.main`	is the plot title
`padjX`	is the minium FDR to highlight as significant
`flip`	when set to TRUE flips the data
`jg.experimentResultsDeseqis`	the result Dataframe from DESeq2 for the experimental conditions

Examples

```
data(jg.controlResultsDeseq,package="Brundle")
data(jg.experimentResultsDeseq,package="Brundle")

jg.plotDeSeqCombined(jg.controlResultsDeseq,
                     jg.experimentResultsDeseq,
                     title.main="ER and CTCF Binding Folding changes on ER treatment",
                     p=0.01,flip=TRUE)
```

`jg.plotMA` *jg.plotMA*

Description

This function plots both the control and experimental data on an MA plot. It also allows for the user to provide a normalisation coefficent for the data.

Usage

```
jg.plotMA(jg.experimentPeakset, jg.controlPeakset, jg.untreatedNames,
  jg.treatedNames, jg.coefficient)
```

Arguments

`jg.experimentPeakset`	is the peakset of the experimental data extracted from a DiffBind ojbect with jg.dbaGetPeakset
`jg.controlPeakset`	is the peakset of the control data extracted from a DiffBind ojbect with jg.dbaGetPeakset
`jg.untreatedNames`	is a list of sample names for the control or untreated conditions
`jg.treatedNames`	is a list of sample samples for the treated conditions
`jg.coefficient`	is a normalisation coefficent for the data that can be generated via the pipeline. Can be set to 1 to view before normalisation.

`jg.plotNormalization` *jg.plotNormalization*

Description

This function allows the user to visualize the normalisation. It is not needed for the pipeline but provides a helpful illustration of the process.

Usage

```
jg.plotNormalization(jg.controlCountsTreated, jg.controlCountsUntreated)
```

Arguments

`jg.controlCountsTreated`	Control counts extracted from the Diffbind object for the treated condition using jg.getControlCounts
`jg.controlCountsUntreated`	Control ounts extracted from the Diffbind object for the untreated condition using jg.getControlCounts

Examples

```
data(jg.controlCountsTreated, package="Brundle")
data(jg.controlCountsUntreated, package="Brundle")
jg.plotNormalization(jg.controlCountsTreated, jg.controlCountsUntreated)
```

`jg.runDeSeq` *jg.runDeSeq*

Description

Runs DESeq2 on our peakset after we have obtained the normalised size factors.

Usage

```
jg.runDeSeq(jg.PeaksetDeSeq, jg.conditions, jg.SizeFactors = NULL)
```

Arguments

`jg.PeaksetDeSeq`	is the experimental peakset formatted for DESeq2
`jg.conditions`	is the list of conditions to be compared
`jg.SizeFactors`	is the size factors generated from the control samples

Examples

```
jg.runDeSeq(jg.PeaksetDeSeq,jg.conditions, jg.SizeFactors = NULL)
```

Index